# Designing a virtual whole body tactile sensor suit for a simulated humanoid robot using inverse dynamics

Salman Faraji* and Auke Jan Ijspeert*

***Abstract*—In this paper, we propose a novel architecture to estimate external forces applied to a compliantly controlled balancing robot in simulations. We use similar dynamics equations used in the controller to find mismatches in the available sensory data and associate them to an unknown external force. Then by decomposing Jacobians, we search over the surface of all body links in the robot to find the force application point. By approximating link geometries with ellipsoids, we can derive analytic solutions to solve the search problem very fast in real time. The proposed approach is tested on a complex humanoid robot in simulations where it outperforms static estimators over fast dynamic motions. We foresee a lot of applications for this method especially in human-robot interactions where it can serve as a whole body virtual suit of tactile sensors. It can also be very useful in identifying the inertial properties of objects being manipulated or mounted on the robot like a backpack.**

## I. INTRODUCTION

In humanoid robotics control, dynamics-based approaches are becoming more and more popular [1]–[3]. Such trend has many motivations behind, especially compliance and precision. Compliance is a crucial feature of a humanoid robot in the tasks that involve human-robot interaction. But going further, compliance can be very useful for the robot as well, mainly in terms of safer operation and avoiding self-damage. Compliance has two different types, active and passive [4]. Passive compliance is mainly inspired by human tendons and their energy storing role in addition to absorbing impacts that often exist in loco-manipulation tasks. However from the control perspective, certain tasks might require different levels of compliance in the task or joint space. Such compliance can not be realized by passive elastic elements that are often in series with actuators [5] in most recent humanoid robots [6]. Here, the other type of compliance becomes very important, i.e. active compliance. The control algorithm takes advantage of sensory data available on the robot to generate actuator policies that behave as if a real spring is there in the robot. Such virtual elastic element [7] can be used in control paradigms such as balancing, manipulation or locomotion tasks.

In this paper, based on our previous works on compliant balancing controllers [1], we propose a complementary architecture that estimates external disturbances as well, using the available sensory data. This architecture is briefly presented in Fig. 1. By fusing available sensory data, we can estimate global states and take advantage of dynamics equations to control the robot. On the other hand, we can also use same equations and dynamics sensory information to identify mismatches and associate them to unknown sources of error. In the following, we first briefly describe the proposed architecture.

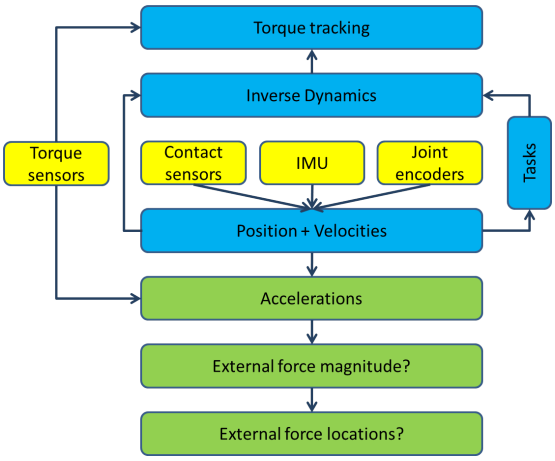


Fig. 1: Control/estimation architecture proposed to estimate external disturbances. The sensory data available from the robot are fused together to estimate the current state of the system, including positions, velocities and accelerations. These states are used to perform desired tasks using inverse dynamics and torque tracking blocks. On the other hand, using the redundancy in the dynamics information available, we can estimate external disturbances and their locations.

### A. Inverse dynamics

As soon as active compliance is considered, we also need to think about available kinematics and dynamics information. Such information can be used in model-based or model-free control approaches that try to perform the desired tasks while providing a certain level of compliance. Generally, kinematics information give the current state of the robot and determine how precise the desired task is being performed. On the other hand, dynamics information can be used to determine interactions and compliance. Among different control approaches, inverse dynamics is becoming very popular as it can deal with the complex nature of humanoids efficiently.

### B. State estimation

The complexity of humanoids can be both due to high number of degrees of freedom and the fact that humanoids are floating based robots. The latter is practically more important in fact, although it seems not if computational power is solely considered. Inverse dynamics requires the full state of the robot including joint configurations and base variables, expressed in the global frame. Since the robot does not have direct measurements of base variables, there is often a

*Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

978-1-5090-3762-9/16/$31.00 

geometric filter block that fuses all available sensory data to determine the full state. Therefore, inverse dynamics is always accompanied by another complex routine that has certain limitations.

### C. Torque tracking

Apart from observing the state, inverse dynamics also needs precise torque tracking implemented in actuator level. Although models are not very accurate, precise torque tracking can still increase consistency of the model with the real robot. This block requires measurements of either direct joint torques, or corresponding series elastic deflections or the electrical current in the actuators. Compared to the traditional position controlled robots [8], [9], torque measurement can provide additional information about interactions or un-modeled phenomena.

### D. Dynamics inconsistency

The idea behind using the model is to generate control policies that realize tasks of certain priorities [10]. The knowledge required for many of these tasks is already available in the model. Balancing task for example is mainly linked to gravity compensation and keeping static stability which are dynamics and kinematics information respectively. Recently, compared to [10], we have developed a different formulation of the inverse dynamics problem which considers the system dynamics, desired tasks and physical limitations altogether to generate proper joint torques [11]. This general and versatile formulation is later combined with state estimation and torque tracking algorithms and successfully tested on our real robot Coman [1]. The very important feature of our implementation is working only with joint torques. We do not put secondary loops over joint velocities or positions that require integration of desired joint accelerations, coming out of inverse dynamics. This simple feature in fact improves precision, Cartesian compliance and stability by avoiding interfering stiffnesses.

Our formulation let us define all the tasks in Cartesian space. Basically, there are certain trajectories considered for the end effectors (hands, feet, CoM and torso orientation) and followed by Cartesian PID regulators that generate accelerations in the end. The notion of active compliance appears here, since our inverse dynamics compromises different tasks by predefined priorities. In case of external perturbation for example, the desired acceleration is nonzero, while the robot does not move anymore (refer to [1], [11] for real demonstrations). In fact, inverse dynamics is blind to the external disturbance and produces a force that reacts to the perturbation, proportional to the displacement.

### E. External force estimation

In the present work, we want to investigate dynamics errors and infer information about perturbations. This is possible only thanks to the three previously mentioned blocks which estimate the global states precisely, plan consistently and execute the plan with fast torque controllers. Thanks to 6D contact force/wrench sensors and all joint torque sensors in our platform, apart from the control, we can re-write the equation of motion with available sensory data. We also take advantage of other kinematic constraints to determine joint accelerations. Now, we can simply investigate if there is any mismatch in the equation of motion and associate it with an external force. Fig. 2 demonstrates a simple scenario which gives intuition on how joint torques and contact forces can determine external force positions.

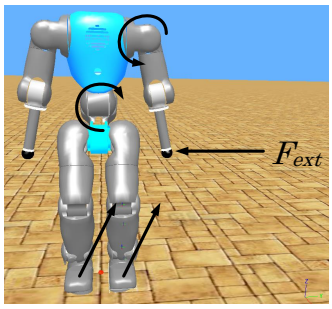


Fig. 2: An intuitive figure, demonstrating how an unknown external force can be estimated based on the internal sensory data. In this case, the lateral force exerted on the hand tip produces an additional torque in all the joints. However, we can only measure the torque around the joint pivot vector in each joint. Therefore in the scenario of this figure, only the shoulder roll sensor detects a larger torque. After the arm, this torque appears in the waist joint as well, going down to the contact 6D force/wrench sensors. The proposed algorithm is limited to a single external force, although it can potentially explore more forces under certain conditions. We also assume a point-wise perturbation without external wrench to simplify the problem.

This idea is not new however and appears in literature, in the context of model identification [12] or impedance control [13]. The main motivation behind looking at the equation of motion is to avoid direct measurement due to many reasons.

- In some cases, there are concerns about pricing, weight or complexity. In robotics arms for example, force estimation by joint torque sensors is popular, either for identification of new objects or for better control performance [13], [14]. These robots are however much simpler in terms of degrees of freedom and more importantly, fixed to a table. Therefore, it might be reliable enough to estimate contact forces by the equation of motion, as state estimation is simpler. Note that this method can be applied to position-controlled robots as well, as far as they have torque sensors.
- On the other hand, it is less practical to put many small sensors together to form a tactile suit with many cables and of course a heavy weight. The artificial skin proposed in [15], [16] is useful however in certain manipulation tasks. The flexible skin proposed in [17] is also very useful in human-robot interaction. However, it is not always easy to anticipate how humans specially elderly and kids would touch the robot.

These shortcomings in the literature of force estimation has motivated us to propose a general approach to estimate external force strengths and locations on the whole body of the robot, without adding any other physical sensor.

The proposed approach uses contact force information to

infer the strength of a single external force and then searches over the surface of all links in the robot to find the best matching correspondence. The first part to find force strengths is inspired by a very simple approach proposed in [18] which uses the second newton law, written for Center of Mass (CoM). For the second part, we propose a similar algorithm used in [19], though computationally more efficient. Searching over all links of the robot basically requires exact geometries of the surfaces. Likar approximates link shapes of an ABB manipulation arm by cylinders [19] and then sets up constrained optimization problems to find the force application point on all cylinders. In our method however, due to the complexity of our humanoid robot in terms of the number of links, we approximate link shapes by ellipsoids which look realistic and propose an alternative exact solution of similar optimization problems. The core idea behind our approach is to break down unknown Jacobians and to find local positions in closed form.

We can cover the whole body of the robot, compared to the limited application of explicit physical touch sensors. We are also not restricted to the end-effectors and basically, search for both forces and application locations over the whole body of the robot. Although the presented work is yet limited to a single external force, it can be very useful in model identification and refinement as well as human-robot interactions, for example detecting pushing or pulling forces. In this paper, we do not use the estimated force to improve the balancing controller, however this can be another application of the proposed method. In next section, we will introduce details of our search algorithm as well as underlying assumptions. Next we will demonstrate the results and discuss the performance and in the end, conclude the paper with proposing possible future extensions.

## II. Methodology

For the purpose of this paper, we base the estimator and simulations upon our previous works [1], [11] where a multi-stage controller performs state estimation and balancing using inverse dynamics. A closer look into these publications will indeed give more insight on the application of the proposed method. Note that in this work, we do not feed the estimated external forces back to the controller to improve the balancing performance.

As mentioned previously, the proposed estimation method has three main stages, acceleration, force strength and force location estimation. The general equations of motion for the whole body of the robot are basically written as:

$$M(q)\ddot{q} + h(q,\dot{q}) = \tau + \sum J^T F \tag{1}$$

Where $q \in \mathbb{R}^{6+23}$ denotes the full state of the robot (including global position and orientation of the base), $M(q)$ denotes mass matrix, $h(q,\dot{q})$ denotes gravitational, centrifugal and Coriolis forces, $\tau$ is joint torques, $J$ denotes translational and rotational contact Jacobians and $F$ denotes contact forces or wrenchs. In this set of equations which follows Kane's convention [20], the first six rows are actuation-less ($\tau_{1..6} = 0$) and describe global frame dynamics of the floating based robot. Apart from dynamics equations, we also have to consider fixed contact constraints:

$$\dot{J}(q)\dot{q} + J(q)\ddot{q} = \ddot{x} \tag{2}$$

Where $\ddot{x}$ represents contact acceleration which is zero in case of static contacts. A more detailed discussion about these constraints can be found in [1].

We have direct measurement of joint torques and contact forces as well as $q$ and $\dot{q}$ for the joints. Recently, we have also developed another states estimation algorithm [1] which serves as the basis of this paper. This algorithm determines full vectors $q$ and $\dot{q}$ for the floating base by fusing all sensory data on the robot and contact constraints. However to estimate additional external forces, we need to know joint accelerations as well. So before mentioning the main force-estimating stages, we introduce another stage to determine joint and global accelerations. In the following, these three stages are described.

### A. First stage: Acceleration estimation

In addition to contact forces $F$ and $M$, joint torques $\tau$, joint positions $q$ and velocities $\dot{q}$, our robot Coman is equipped with an Inertial Measurement Unit (IMU) on the pelvis which provides linear accelerations and angular velocities as well as orientations, internally calculated in the IMU by fusing other two variables. Although most of the sensory information is used in the underlying state estimation [1], we use dynamics information (forces, torques and accelerations) again in this stage to find the full vector $\ddot{q}$. As a rough estimation of joint accelerations, we differentiate joint velocities to obtain $\ddot{q}_{opt}$. Coman has optical encoders to measure the velocity directly. However this differentiation is not accurate, because the velocity is measured before the series elastic elements, not after on the link side. We similarly differentiate the gyro velocity to obtain an estimation of base's angular acceleration $\ddot{q}_{gyr}$. A quadratic optimization problem is then formulated to fuse these estimations with IMU accelerations $\ddot{q}_{IMU}$, the equation of motion (1) and contact constraints (2).

$$\begin{aligned}
\min_{\ddot{q}} \sum \mathbf{V_{Q_{\delta_E}}}(\delta_\mathbf{E}) + \mathbf{V_{Q_{\delta_R}}}(\delta_\mathbf{R}) + \mathbf{V_{Q_{\delta_I}}}(\delta_\mathbf{I}) + \sum_{\mathbf{i=1}}^{\mathbf{n_c}} \mathbf{V_{Q_{\delta_i}}}(\delta_\mathbf{i}) \\
M(q)\ddot{q} + h(q,\dot{q}) = \tau + \textstyle\sum_{i=1}^{n_c} J_i^T F + \delta_E \\
\ddot{q}(1..3) = \ddot{q}_{IMU} + \delta_I \\
\ddot{q}(4..6) = \ddot{q}_{gyr} + \delta_G \\
\ddot{q}(7..29) = \ddot{q}_{opt} + \delta_O \\
\dot{J}_i(q)\dot{q} + J_i(q)\ddot{q} = \ddot{x}_i + \delta_i, i = 1..n_c
\end{aligned} \tag{3}$$

Here, we define $V_Q(\psi) = \psi^T Q \psi$ to represent a quadratic function. The role of slack variables $\delta$ in this formulation is to create flexible fusion of sensory data and reject noises, similar to Kalman filtering. In fact optimization approaches are becoming more popular for filtering sensory data in humanoid robots [21] as they provide the option to add inequality constraints, compared to the conventional Kalman filtering [22]. It is also easy to prove that such optimization is equivalent to

Kalman filtering, if the quadratic cost matrices are equal to the inverse of covariance matrices in Kalman formulation. In our implementation however we tune these matrices manually and keep them constant. Although we lose optimality, the performance still remains acceptable. The choices for positive definite cost matrices are heuristically diagonal-shape with the following numbers $Q_{\delta_E} : 1$, $Q_{\delta_I} : 10^4$, $Q_{\delta_G} : 1$, $Q_{\delta_O} : 1$ and $Q_{\delta_i} : 10^4$ to ensure constraint satisfaction and give more importance to the IMU accelerations than velocity differentiations. This quadratic problem is solved in less than $100\mu s$ by CVXGEN QP optimizer [23]. Note also the structure of our generalized floating base coordinate vector where the first three elements are global translation (of the pelvis), the second three are global rotation (of the pelvis) and the rest are joint coordinates.

### B. Second stage: Force strength estimation

In this work, we limit the formulation to detection of a single external force, although it can be easily extended to detect an additional external moment at the same time. The strength of external force can be simply calculated by looking at the first three lines of the equation of motion. Since in the previous stage, more importance is given to the IMU reading rather than the equation of motion, the vector $\delta_E$ estimates the projection of external force onto the space of floating base generalized coordinates. Regarding the structure of our state vector, the first three elements of $\delta_E$ indeed estimate the external force vector itself. Note that the vector $\delta_E$ can already be used in the balancing controller to improve the performance without performing the next stages of this estimator. This is more generic without the limiting assumptions of single force or ellipsoid geometry. In the present work however we do not explore this possibility and leave it for future works. Note also that in case of static balance, all accelerations are zero and the external force can be calculated in a simpler way. However we prefer to keep the mass matrix (and the first stage accordingly) in order to explore dynamic motions. Disabling the first stage is indeed equivalent to a static algorithm which is compared with the full estimator later in the results section.

### C. Third stage: Force location estimation

Once the extra force is calculated, we can consider the equation of motion again. In fact the resulting $\delta_E$ in the first stage accounts for an unknown external force $F_{ext}$ that is applied to an unknown point $P$, located on an unknown body link $B_i$ of the robot:

$$\delta_E = J_P^{v^T} R_{B_i}^T F_{ext} = J_P^{v^T} \hat{F}_{ext} \tag{4}$$

Where $R_{B_i}$ denotes body's rotation matrix. Note that all Jacobians are originally written in the local frame in Kane's formulation. Therefore we have to transform the globally expressed $F_{ext}$ back to the local frame of $B_i$ to obtain $\hat{F}_{ext}$.

In the following, we are proposing a procedure to find the body index $i$ and the force location $P$. Note that $P$ is expressed in the local frame of the body $B_i$. Therefore, we do not exactly know the Jacobian $J_P^v$ unlike many other approaches

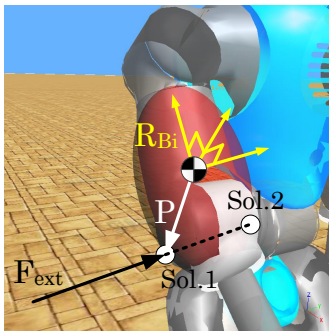


Fig. 3: Demonstration of the upper arm of the robot, approximated with an ellipsoid of proper size. The local frame $B_i$ is placed on the CoM of the link, which we consider the centroid of the ellipsoid as well. When an external push is applied on the link at a local point $P$, the algorithm basically intersects a line calculated based on the measured dynamics variables with all the ellipsoids over the whole body and then selects the one making the minimum cost.

that assume known Jacobian [14]. Fig. 3 demonstrates $B_i$, the exact link shape, approximate ellipsoid and the unknown point $P$. Considering the frame $B_i$ located on the CoM of the body link $B_i$, we can use known Jacobians to find the translational velocity of the point $P$:

$$\begin{aligned} x_P &= x_{B_i} + R_{B_i} P \\ \dot{x}_P &= R_{B_i} J_P^v \dot{q} = R_{B_i}(v_{B_i} + S(\omega_{B_i})P) \\ v_{B_i} &= J_{B_i}^v \dot{q} \end{aligned} \tag{5}$$

Where $S(\psi)$ represents skew-symmetric matrix, obtained from the vector $\psi$. Similarly, we can find the angular velocity of the body $B_i$ on which the point $P$ is located:

$$\begin{aligned} R_P &= R_{B_i} \\ \omega_{B_i} &= J_P^\omega \dot{q} = J_{B_i}^\omega \dot{q} \end{aligned} \tag{6}$$

Now, we can link the two sets of equations together:

$$\begin{aligned} R_{B_i} J_P^v \dot{q} &= R_{B_i}(J_{B_i}^v \dot{q} + S(\omega_{B_i})P) \\ &= R_{B_i}(J_{B_i}^v \dot{q} - S(P)\omega_{B_i}) \\ &= R_{B_i}(J_{B_i}^v \dot{q} - S(P) J_{B_i}^\omega \dot{q}) \end{aligned}$$

Therefore, the Jacobian of the point $P$ can be written as:

$$J_P^v = J_{B_i}^v - S(P) J_{B_i}^\omega \tag{7}$$

Which is written in terms of known Jacobians $J_{B_i}^v$ and $J_{B_i}^\omega$ for the $B_i$ frame. However we still need to determine the vector $P$ to obtain the full Jacobian. Coming back to the equation (4), we can write:

$$\begin{aligned} \delta_E = J_P^{v^T} \hat{F}_{ext} &= (J_{B_i}^v - S(P) J_{B_i}^\omega)^T \hat{F}_{ext} \\ &= J_{B_i}^{v^T} \hat{F}_{ext} - J_{B_i}^{\omega^T} S(P)^T \hat{F}_{ext} \\ &= J_{B_i}^{v^T} \hat{F}_{ext} - J_{B_i}^{\omega^T} S(\hat{F}_{ext}) P \end{aligned} \tag{8}$$

As a result, we can obtain a relation for $P$:

$$S(\hat{F}_{ext}) P = -(J_{B_i}^{\omega^T})^+ (\delta_E - J_{B_i}^{v^T} \hat{F}_{ext}) \tag{9}$$

where $()^+$ denotes pseudo-inversion and $J_{B_i}^\omega$ and $J_{B_i}^v$ are known. Although there are three equations in (9), due to rank

deficiency of skew symmetric matrices, we effectively have two equations with three unknown components of $P$.

Since we also know that the force application point is located on the body surface of the robot, we have to intersect the resulting line obtained from (9) with the surface of each link $B_i$ to obtain two points maximally, in case of convex link geometry. Our proposed method however approximates each link with an ellipsoid which matches the actual link shape as much as possible. Such ellipsoid is formulated around the CoM of $B_i$:

$$\frac{P(1)^2}{a^2} + \frac{P(2)^2}{b^2} + \frac{P(3)^2}{c^2} - 1 = 0 \tag{10}$$

Where semi-principal axes have length of $a$, $b$ and $c$, proportional to the diagonal elements of the inertia tensor for each link with slight hand tunings. Our geometric model is based on very precise CAD models of the robot [6]. In future however, we plan to setup optimization routines to adjust the positioning and scaling of the ellipsoids for better matching.

Now, we have the third equation to find all components of the variable $P$. A simple approach is to take two variables out of (9) in terms of the third variable, replace them in (10) and solve a polynomial of degree two. In the end, depending on the polynomial, we might have zero, one or two solutions. In case of having no solution, we simply set the discriminant of the second-degree equation to zero to obtain two points close to the ellipsoid. The proposed search procedure takes the following steps:

1) Calculate the full acceleration vector $\ddot{q}$ through the first stage.
2) Extract the external force vector $F_{ext}$ out of the first three lines of (1).
3) Search over all body links $B_i$: solve the closed form equations for $P$.
4) Calculate the norm of the error vector $e1 = \delta_E - J_P^T \hat{F}_{ext}$.
5) Calculate the distance of $P$ to the ellipsoid by taking the left hand side of equation (10), $e2 = lhs(10)$.
6) Pick the link with minimal error $e1 + e2$.

This procedure is in fact very fast compared to optimization based approaches proposed in [19]. All the stages together can run in real-time ($100\mu s$) on a moderate Core i5 CPU. The algorithm can also be extended to other geometric shapes easily as the idea of intersecting lines with the volumes is the same. The key point to obtain closed form solutions is indeed to decompose the Jacobian into known and unknown parts. In the next section, we demonstrate simulation results over different tasks, followed by discussions on the performance and generality of this approach.

## III. RESULTS AND DISCUSSIONS

As explained, at each time step, the search process determines the force vector, body link and the local point $P$ where the external force is applied. In this implementation, we do not set up specific time-filtering algorithms to deal with noises and other uncertainties intrinsically existing in our search process. Instead, we rely on our state estimation methods to provide clean and stable fusion of sensory data. It should also be noted that in practice, out of the two solutions obtained for the best link, we only consider the one corresponding to a pulling force, i.e. the point where the external force vector goes out of the body link. On the real robot however, the other point should be selected as disturbances are mainly pushing forces, unless someone can pull the robot.

### A. First scenario: Exploring different body links

The first scenario is characterizing the resulting precision for all the links of the robot. Here, we apply forces to various links of the robot, while performing stable and compliant balancing task. We let the robot stabilize and demonstrate the final force estimation and the corresponding body link in Fig. 4. One can obviously see that in static situations, the algorithm finds an acceptable estimate of the external force, although sometimes the body link is very narrow like the lower arm. In these cases, the intersection is hard to find, regarding uncertainties available in the procedure. The estimated force application points $P$ are acceptable too, although there is a shift sometimes due to our state estimation method or passive elastic elements (joint springs). Since on the real robot, link-side encoders (after springs) do not have enough resolution, we always use motor-side encoders to build kinematic chains. In the simulations of this work also, we follow the same convention to be more realistic.

Although the precision is convincing, there are situations where the force can not be uniquely estimated. For example imagine in Fig. 4, there is an external force applied to the tip of the hand, directly upwards. In this case, the algorithm might report any of the links in the arm, as the external force produces the same torque in the shoulder joint. Finding a unique solution is even more challenging in dynamic motions, depending on the force magnitude, agility of the task, closeness of the force to the joint pivot and etc. The external force vector can always be estimated, i.e. the second stage is robust as long as being provided with correct accelerations. However, there should be another high level method that decides whether the reported link and $P$ variables are correct or not. We leave the design of such complex method for future works. However in this paper, we would like to characterize the performance and argue about challenging parts of this algorithm.

### B. Second scenario: Exploring force directions

In the second scenario, we investigate the effect of force direction and keep the body link choice fixed. Fig. 5 demonstrates a scenario where different pulling forces are applied to the left thigh of the robot. Detecting forces on the two legs is also challenging as there is redundancy in the loop created when both feet are supporting a portion of the weight. One can observe that in Fig. 5, there is only one case where the algorithm reports a different body link, i.e. shank. This can be due to many reasons such is inaccuracy of ellipsoidal approximations, redundancy problem or alignment of the external force with the shank. However, it is promising that the

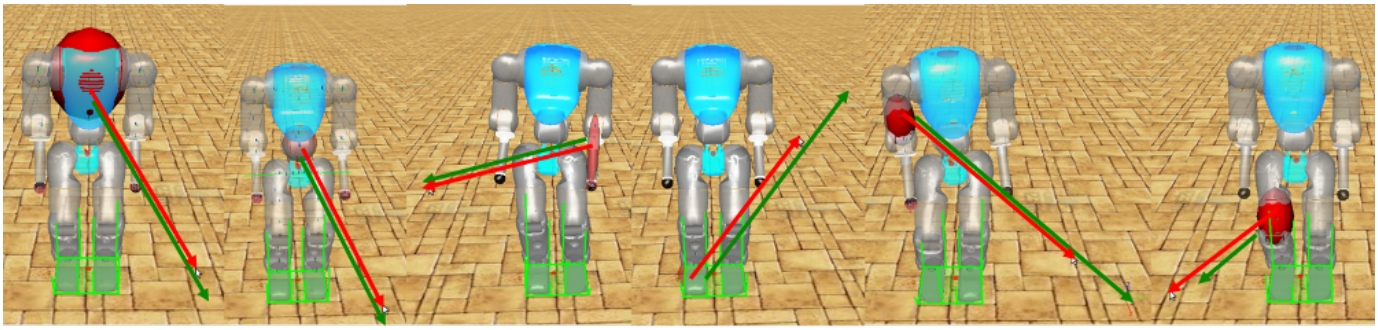

Fig. 4: Few positions where the balancing robot has stabilized after being pulled by an arbitrary external force. In this figure, red arrows are pulling forces applied manually in the user interface of our simulation software, while green arrows show the final estimation. Ellipsoids are transparently shown in red, demonstrating the body link subject to the pull, determined by the algorithm. One can verify that arrows have minimal shift while the estimated body links are matching the actual links (where the red arrows originate from). In general, detection of a force applied to the upper links is easier as it influences the torque in many joints down to the foot. Also, detecting forces on bigger links is easier too, since finding an intersection is more probable. Therefore, one can deduce that the forces being applied to the torso are the easiest to estimate.

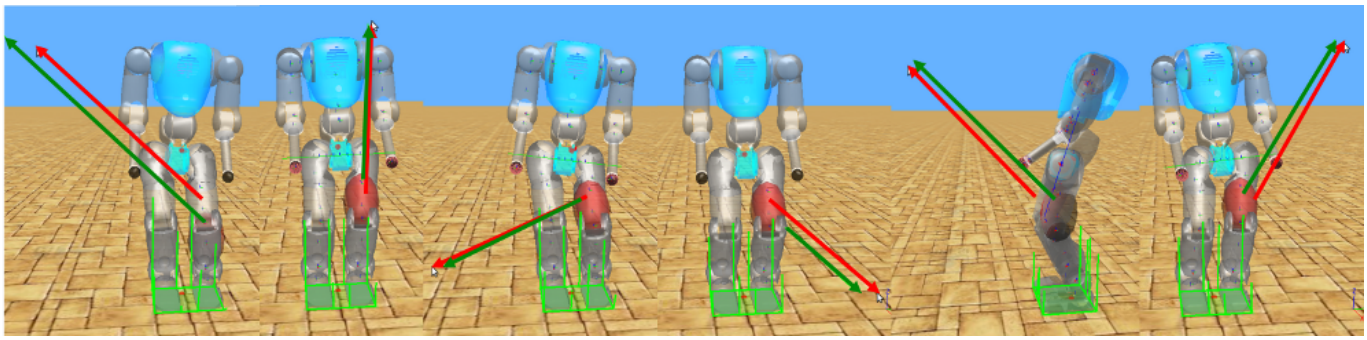

Fig. 5: Demonstration of different pulling forces applied to the left thigh. Red arrows are actual force vectors while green arrows demonstrate the estimated force. Although the matching is good in most of the cases, the left robot can not estimate the body link correctly.

reported body link is the shank, close to the original thigh link.

### C. Third scenario: Dynamic motion

An alternative to the proposed method is to completely remove accelerations from the equation of motion, assuming that the robot is static. In this case, we can disable the first stage of the algorithm as it is not required anymore. This alternative is of course simpler and less demanding in terms of online calculations, however it only works when the robot motion is very slow. Remember that there are uncertainties in the model, state estimation, geometry approximation and etc. If these uncertainties are dominant, i.e. their magnitude is larger than the external force, the algorithm might jump between different body links and provide a less reliable estimation. Therefore, we are interested to know whether accelerations can help in dynamic motions or not. In other words, the first stage of the estimator is supposed to compensate dynamic effects so that the error vector $\delta_E$ merely depends on the external force and uncertainties.

To investigate the role of accelerations, we consider a second version of the estimator where accelerations are set to zero. Such static estimator is compared with the original one over some dynamic motions in Fig. 7. The robot is performing up and down motions with turning around the yaw

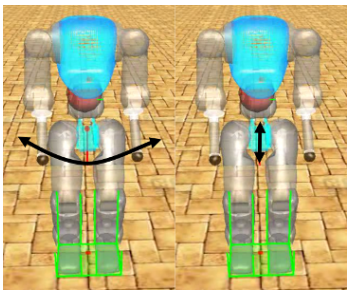

Fig. 6: In the dynamic scenario, the robot is performing a combination of two different tasks at the same time: following a sinusoidal trajectory of $15cm$ up and down and turning around the yaw axis with the same regime for 50 degrees.

axis, demonstrated in Fig. 6. Meanwhile, we apply different forces and compare the estimated values coming out of the two static and dynamic variations. As expected, it is obvious that the dynamic formulation performs better in estimating the force magnitude, even in the absence of external force where the magnitude is zero. It can also provide a more stable body link estimation, although it fails over the course of last perturbation where the ellipsoid is too narrow to find intersections. Overall, accelerations are helping to keep consistency and compensate the fast motions of the robot.

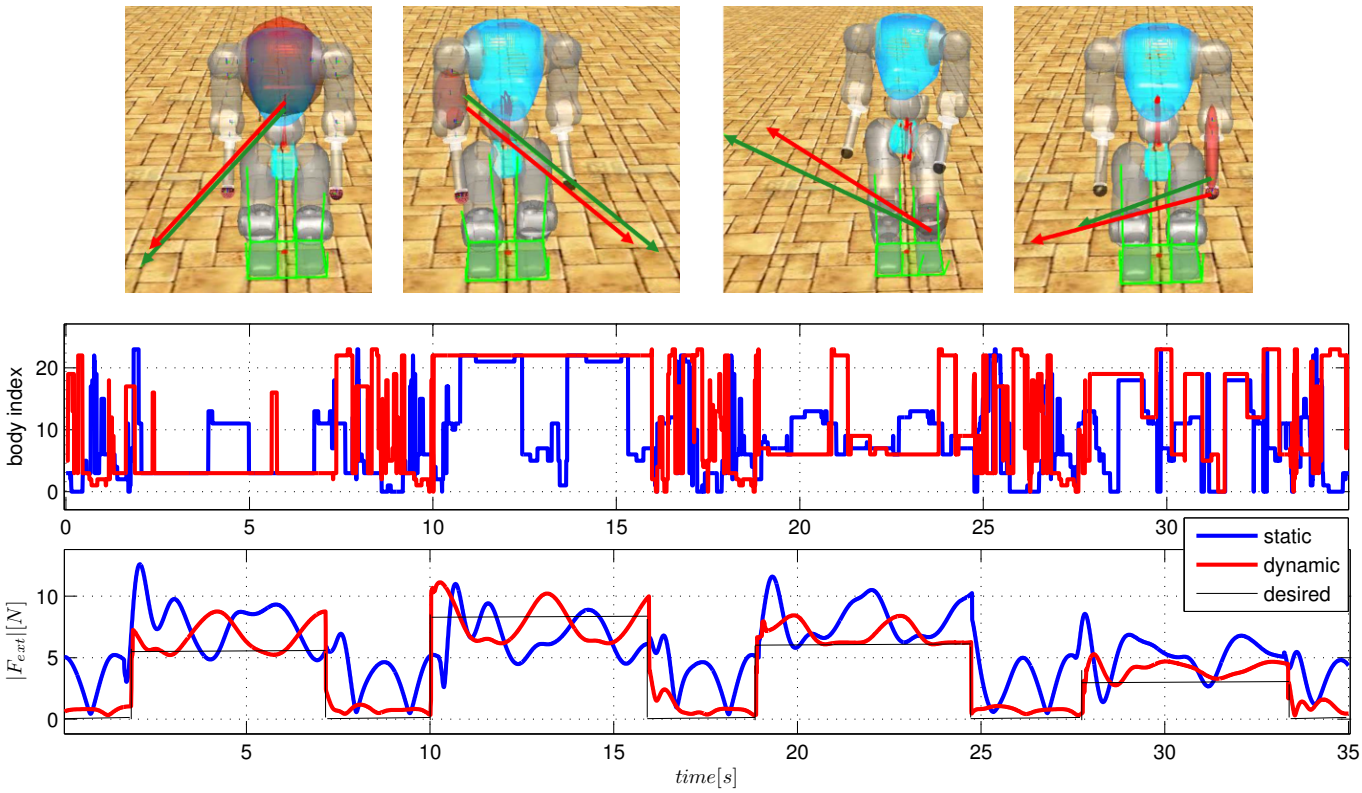


Fig. 7: Comparing performance of the full algorithm in dynamic motions with the case where the first stage is turned off. In this case, accelerations are zero and the algorithm is static. Here, we apply four forces to different body links and plot the estimated force magnitude and the body index. The geometry of the force is also shown on top of the curves. It is obvious that the dynamic estimator performs better, at least over the first three perturbations. It stably reports the body index and rarely jumps to another value. The last pull is more challenging to detect however, as the ellipsoid is too narrow to find intersections. Note that the curves are not meaningful when no force is applied, though the dynamic estimator can estimate the force magnitude (which is zero) better than the static estimator. Full demonstrations are available in the multimedia attachment.

Therefore, the proposed algorithm performs better than static formulations, with minimal computational requirement.

In case of small force magnitudes or very fast motions, another high level algorithm is needed to threshold the estimated force and filter out jumps in the body index. Adding more damping to the controller will also reduce these jumps. The required precision of course depends on the final application of the proposed method. In case of human-robot interaction or model identification, indeed a statistically correct estimation over a certain period of time is expected and occasional jumps are tolerable. In an online control however, one should carefully filter these jumps to avoid instability.

## IV. CONCLUSION

In this paper, we proposed an estimation architecture of multiple stages to determine the global state of the robot, accelerations and external disturbances at the same time. We fuse available sensory data to calculate global states and from the mismatch observed in the equation of motion, we estimate the external force vector. Then we search over all body links to find the point where this external force is applied. The procedure requires knowledge about link geometries which we approximate by ellipsoids to reduce the burden of calculations. Although the proposed method requires an optimization for determining accelerations, in the next stages, the closed form solution helps speeding up the search process.

We characterized the estimation precision for different body links, force directions and dynamic motions. The algorithm proves to provide acceptable results for rather small forces (equivalent to $0.5-1kg$ compared to the weight of robot which is about $30kg$). This is promising for application on the real robot although a higher level filter is required. The process is in fact challenging due to model mismatches, state estimation inaccuracies, ellipsoid approximations, singularities, redundancies and the noise. In this paper, we tried to implement the algorithm as realistic as possible, simulating conditions on the real robot. The proposed algorithm has certain advantages over other similar works:

- Estimating force magnitude and direction.
- Finding force application location.
- Working in dynamical motions as well.
- Computationally very fast.
- Handling floating based calculations.
- Handling many degrees of freedom.

However currently, the method is limited to detection of a single external force. In future, we would like to extend the method to detect external moments and possibly multiple forces. Detecting a single moment is rather straightforward

though, since it can be simply detected from the contact wrenches reported by the sensors. Considering exact link geometries as well as testing the method on the real robot are also part of our future work. Please refer to the multimedia attachment for full demonstrations of the scenarios discussed in this paper.

## V. ACKNOWLEDGMENTS

This work was funded by the WALK-MAN project (European Community's 7th Framework Programme: FP7-ICT 611832).

## REFERENCES

[1] S. Faraji, L. Colasanto, and A. J. Ijspeert, "Practical considerations in using inverse dynamics on a humanoid robot: torque tracking, sensor fusion and cartesian control laws," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1619–1626.

[2] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," in *submitted to Intelligent Robots and Systems, 2014. IROS 2014. IEEE/RSJ International Conference on*.

[3] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2589–2594.

[4] W. Wang, R. N. Loh, and E. Y. Gu, "Passive compliance versus active compliance in robot-based automated assembly systems," *Industrial Robot: An International Journal*, vol. 25, no. 1, pp. 48–57, 1998.

[5] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1995, pp. 399–406.

[6] L. Colasanto, N. G. Tsagarakis, and D. G. Caldwell, "A compact model for the compliant humanoid robot coman," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*. IEEE, 2012, pp. 688–694.

[7] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The I. J. of Robotics Research*, vol. 26, no. 2, pp. 129–143, February 2001.

[8] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3. IEEE, 2002, pp. 2478–2483.

[9] F. L. Moro, N. G. Tsagarakis, and D. G. Caldwell, "A human-like walking for the compliant humanoid coman based on com trajectory reconstruction from kinematic motion primitives," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 364–370.

[10] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 981–988.

[11] S. Faraji, S. Pouya, and A. Ijspeert, "Robust and agile 3d biped walking with steering capability using a footstep predictive approach," in *Robotics Science and Systems (RSS)*, 2014.

[12] S. Traversaro, A. Del Prete, R. Muradore, L. Natale, and F. Nori, "Inertial parameter identification including friction and motor dynamics," *arXiv preprint arXiv:1410.4410*, 2014.

[13] D. P. Le, J. Choi, and S. Kang, "External force estimation using joint torque sensors and its application to impedance control of a robot manipulator," in *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*. IEEE, 2013, pp. 1794–1798.

[14] A. Colomé, D. Pardo, G. Alenya, and C. Torras, "External force estimation during compliant robot manipulation," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3535–3540.

[15] M. Mistry and L. Righetti, "Operational space control of constrained and underactuated systems," *Robotics: Science and systems VII*, pp. 225–232, 2012.

[16] M. Strohmayr and D. Schneider, "The dlr artificial skin step ii: Scalability as a prerequisite for whole-body covers," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4721–4728.

[17] T. Noda, T. Miyashita, H. Ishiguro, and N. Hagita, "Super-flexible skin sensors embedded on the whole body, self-organizing based on haptic interactions," *Human-Robot Interaction in Social Robotics*, p. 183, 2012.

[18] K. Kaneko, F. Kanehiro, M. Morisawa, E. Yoshida, and J.-P. Laumond, "Disturbance observer that estimates external force acting on humanoid robots," in *Advanced Motion Control (AMC), 2012 12th IEEE International Workshop on*. IEEE, 2012, pp. 1–6.

[19] N. Likar and L. Zlajpah, "External joint torque-based estimation of contact information," *International Journal of Advanced Robotic Systems*, vol. 11, 2014.

[20] T. R. Kane and D. A. Levinson, "Multibody dynamics," *Journal of applied Mechanics*, vol. 50, no. 4b, pp. 1071–1078, 1983.

[21] X. Xinjilefu, S. Feng, and C. G. Atkeson, "Dynamic state estimation using quadratic programming," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 989–994.

[22] G. Welch and G. Bishop, "An introduction to the kalman filter," 1995.

[23] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.